Assignment 02 – GPA calculator

# Comp 123 – Programming 2

**DUE:** Friday, 25 June 2021, 23:59:59, Eastern time

**VALUE:** 10%

## Overview

Create a console app to display all students' calculated GPAs, or an individual student's course grades and GPA. After displaying the selected data, the user will be prompted with the same choices again.

The various letter grades and number ranges are included in this chart, along with the GPA equivalent.

|  |  |  |
| --- | --- | --- |
| **Letter grade** | **Number grade range** | **GPA** |
| A+ | 90 - 100 | 4.5 |
| A | 80 - 89 | 4.0 |
| B+ | 75 - 79 | 3.5 |
| B | 70 - 74 | 3.0 |
| C+ | 65 - 69 | 2.5 |
| C | 60 - 64 | 2.0 |
| D+ | 55 - 59 | 1.5 |
| D | 50 - 54 | 1.0 |
| F | 0 - 49 | 0.0 |

Serialize the student data in a file. If no file is found at startup, add the provided seed data (see **Assignment02\_seedData.txt**). As students are added, persist all student data to a local file. Your application will access this student data from a StudentRepository class.

The application will not support adding or editing students via the prompt. If the input value provided is not valid (student number is not in the StudentRepository; input is not the option for showing all students' GPA, or to exit), the application will inform the user of that by writing a message to the console and showing the options again.

Some sample outputs are provided in **Assignment02\_sampleOutputs.txt**.

## Requirements

1. Implement the following class diagrams and NOTHING additional:  
   Graphical user interface, application, chat or text message

   Description automatically generated
2. Write these methods and properties based on the descriptions provided:
   1. GradeUtils.ToLetterGrade(double) converts a number grade to a letter grade.
   2. GradeUtils.LetterToGpa(string) converts a letter grade to the GPA equivalent.
   3. GradeUtils.GpaToLetter(double) converts a GPA value to the letter equivalent.
   4. Student.AverageGpa() converts each Grade.LetterGrade in Student.Grades to a GPA value and averages the result.
   5. Student.GPA does not have a setter, and the getter should return Student.AverageGpa() to get the student's GPA similar to this: public double Gpa => AverageGpa();
3. Write the StudentRepository so that:
   1. It implements IRepository.
   2. Other classes do not have direct access to the Student collection.
   3. The LoadRepository() and SaveRepository() are called by the class as needed to ensure data integrity and availability. You should need to use LoadRepository() only once when running the application.
   4. You won't need to call AddStudent(), however it must be coded so that it will correctly add data to the repository and serialize the repository to the file.
4. Implement the Program class so:
   1. it accesses the student repository only through the \_studentRepository property, which is instantiated only once.
   2. IsValidInput() determines if the user input is acceptable. It does not take action on invalid inputs.
   3. HandleInput() handles valid user inputs.

## Submitting your work

Code will be submitted via the eCentennial assignment folders **Assignment 02**. All project files must be contained in a zip file named following the schema below, replacing **mmacdonn** with your Centennial College user ID (your email address, without **@my.centennialcollege.ca**).

Schema: **Comp123-mmacdonn-Assign02.zip**

Only one file in a folder will be accepted for grading. In the event more files are submitted, **the top-most zip file matching the appropriate naming schema** will be the only file checked for grading **regardless of additional notes in the file name and submission time**. It is **highly recommended** you remove any files other than the desired submission.

Please be aware that you may be required to demonstrate your submission at any time.

## Late submissions

The folder will close on a timer set for the above time. No materials, nor changes, will be accepted after that time.

In the event of extenuating circumstances, extensions will be considered on a case-by-case basis. Additional information will be requested in such a situation.

## External code and tutoring

Software engineering is an inherently collaborative and open discipline. However, as an academic institution it is important to identify the individual's own knowledge and understanding, as well as adhering to [Academic Honesty](https://www.centennialcollege.ca/studenthub/student-services/advising-and-academic-resources/advising-services/academic-honesty/) and [Integrity](https://library.centennialcollege.ca/help-services/research-help/academic-integrity/). For this reason, the use of sources outside the lecture and course materials are not permitted.

Additionally, discussion of assignments with your peers or tutors is permitted. However, all materials must be created by the individual student, and not through the actions or aid of another person, whether they are associated with Centennial College or not.

Should you be interested in alternative solutions, you are encouraged to research them and discuss them with the professor. You must do so **before** including such materials in any assignments, graded work, or tests. Failure to do so may result in a violation of Centennial College's [Academic Honesty and Plagiarism Policy](https://p.widencdn.net/7o6xrb/AcademicHonestyAndPlagiarismPolicy).